8-2019

Improving Optimization of Convolutional Neural Networks through Parameter Fine-tuning

Nicholas C. Becherer

John M. Pecarina

Scott L. Nykl
Air Force Institute of Technology

Kenneth M. Hopkinson
Air Force Institute of Technology

Recommended Citation

Becherer, N., Pecarina, J. M., Nykl, S. L., & Hopkinson, K. M. (2019). Improving Optimization of Convolutional Neural Networks through Parameter Fine-tuning. Neural Computing and Applications, 31(8), 3469–3479. https://doi.org/10.1007/s00521-017-3285-0

Neural Computing and Applications (2019) 31:3469–3479
https://doi.org/10.1007/s00521-017-3285-0

ORIGINAL ARTICLE

Improving optimization of convolutional neural networks through parameter fine-tuning

Nicholas Becherer[1] · John Pecarina[1] · Scott Nykl[1] · Kenneth Hopkinson[1]

Received: 16 May 2017 / Accepted: 13 November 2017 / Published online: 25 November 2017

Abstract
In recent years, convolutional neural networks have achieved state-of-the-art performance in a number of computer vision problems such as image classification. Prior research has shown that a transfer learning technique known as parameter fine-tuning wherein a network is pre-trained on a different dataset can boost the performance of these networks. However, the topic of identifying the best source dataset and learning strategy for a given target domain is largely unexplored. Thus, this research presents and evaluates various transfer learning methods for fine-grained image classification as well as the effect on ensemble networks. The results clearly demonstrate the effectiveness of parameter fine-tuning over random initialization. We find that training should not be reduced after transferring weights, larger, more similar networks tend to be the best source task, and parameter fine-tuning can often outperform randomly initialized ensembles. The experimental framework and findings will help to train models with improved accuracy.

Keywords Convolutional neural networks · Transfer learning · Computer vision · Parameter fine-tuning

1 Introduction

Convolutional neural networks (CNNs) are machine learning models that extend the traditional artificial neural network by adding increased depth and additional constraints to the early layers. Recent work has focused on tuning their architecture to achieve maximum performance on benchmarks such as the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [1, 2].

CNNs are not a new topic in the field of computer vision. They can trace their origins back to the early 1980s with Fukushima's Neocognitron [4]. More directly, they were shown to be highly effective in the 1990s when used for handwritten digit recognition and eventually in industry for automated check readers [5, 6]. They rely on several successive convolutional layers to extract information from an image. Since convolution is a shift and slide operation, it is invariant to translations in the data. Most importantly, these convolutional layers are fully learnable through the backprop algorithm, meaning they can identify low- and high-level patterns through supervised training [7]. However, they fell out of favor in the new millennium because of their difficulty scaling to larger problems [8]. Problems beyond the optical character recognition or low-resolution imagery were either too computationally expensive or lacked enough training data to avoid overfitting.

Recently, they have stepped back into the spotlight as these problems have been overcome. In 2012, Krizhevsky et al. [1] leveraged several recent advances to overcome these issues in the 2012 ILSVRC. First, they used NVIDIA's CUDA programming language to implement their CNN on a highly parallel GPU, reducing run time by orders of magnitude [9]. Second, the ImageNet competition included a dataset on the scale of millions of images automatically sourced from the Internet [10]. Combined with several new techniques such as dropout regularization [11] and simple data augmentation, they presented a model dubbed AlexNet that won the competition. Since the

✉ John Pecarina
john.pecarina@us.af.mil

Nicholas Becherer
nbechere@citadel.edu

Scott Nykl
scott.nykl@afit.edu

Kenneth Hopkinson
kenneth.hopkinson@afit.edu

[1] Air Force Institute of Technology, Dayton, OH 45433, USA

introduction of AlexNet, the winning entries for the ImageNet competition have all been CNNs [2, 12, 13]. These newer CNNs have largely advanced the field by making the basic CNN architecture deeper. The Oxford Visual Geometry Group's Visual Geometry Group (VGG) network experimented with 11–19 learnable weight layers, finding that 19 was the optimal architecture [13]. The current leading network, GoogLeNet, has 6.7 million learnable weights across 22 layers [2]. Others have focused on improving the performance of CNNs through data augmentation and training techniques [14]. Yet ultimately, all techniques still required large amounts of training data to be effective.

The issue compelling the need for large amounts of data is due to the fact that CNN training is an extremely complex optimization problem. They typically use stochastic gradient descent (SGD) to find the minima for a loss function and this technique uses large labeled training datasets to minimize effectively. Since SGD is a greedy method, it is not guaranteed to find the global minima. This means that initialization can have an effect on the final outcome. These weights are usually initialized by sampling from a Gaussian distribution [15]. However, it has been shown that a transfer learning technique known as parameter fine-tuning can improve the performance of a CNN compared to random initialization (sometimes to a substantial degree) [3].

However, there is a transfer learning technique that can help overcome a lack of training data. Parameter fine-tuning is a method wherein a network is pre-trained on a different data set and then retrained on the target set. Research has shown this method boosts the performance of a network over random initialization [3]. However, research on this topic is still relatively scarce. This paper will present several gaps in our understanding as well a framework for investigating them. We investigate three main areas. The first is the optimal learn rate for transferred layers. The second is to analyze how different source datasets affect the outcome on a number of target datasets. Lastly, we compare an ensemble of fine-tuned networks to an ensemble of randomly initialized networks. These experiments are done through an experimental framework that allows a single variable to be manipulated and studied across several datasets. Lastly, we apply this methodology to a unique dataset aggregated from several sources and present the results.

This paper is organized as follows. Section 2 describes related work in the area of transfer learning for CNNs, and Sect. 3 exposes our experimental framework. In Sect. 4, we investigate the claim made by Girshick [18] that the reduction in the learning rate may improve the accuracy of the transfer task. We evaluate the use of various source data sets in Sect. 5, and in Sect. 6 we explore the use of ensembles of transfer networks. A conclusion discussion and suggestion of future work is described in Sect. 7.

2 Related works in transfer learning for CNNs

Transfer learning is the study of using data gained from one problem in machine learning and applying it to another related, yet different, problem. With CNNs, there are two main ways to apply transfer learning. The first is to remove the output layer of a trained network and use the raw output of the previous fully connected layer as a generic feature vector that describes a particular image. These features are then used in a number of algorithms which were originally designed around using SIFT or SURF features [16, 17]. Since CNNs are inherently only capable of image classification, extra algorithmic work is necessary to apply it to another problem. The second transfer learning technique is known as parameter fine-tuning, detailed above. This is the focus of this paper.

After the publication of Krizhevsky's AlexNet, Girshick et al. demonstrated both kinds of transfer learning in a single paper [18]. They fine-tuned AlexNet trained on the ImageNet dataset to target the 2007 PASCAL VOC dataset. They note that fine-tuning significantly increases performance; however, they do not report any numbers. One particular interesting claim made is that they reduce the learning rate during fine-tuning in order to avoid 'clobbering the initialization' of the original CNN. They provide no further reasoning or evidence that this is optimal. This paper investigates this claim and finds it to be suboptimal. After fine-tuning, they then pass regions of the image through their CNN and classify the region with SVMs based on the raw output of the CNN. Their methodology is extremely similar to an algorithm that relies on SIFT features, but outperforms it [19]. Girshick et al. [18] are among the first to demonstrate that CNNs can be used effectively for other computer vision problems.

The work of both Razavian et al. [20] and Azizpour et al. [21] also focuses on applying CNNs to other problems. In general, they find that CNNs coupled with SVMs provide competitive results to existing state-of-the-art solutions for many datasets. The paper from Azizpour in particular identifies all the factors involved in transfer learning. However, they only test two different source datasets which are similar in nature. This paper provides and executes an experimental framework for testing different source networks on several different target datasets.

Research from Yosinski et al. [3] focuses exclusively on parameter fine-tuning. Yosinski et al. randomly divide the ImageNet dataset in half. For each half, they train on one half and then target the second half. In both cases, the fine-

tuned network outperformed the randomly initialized network. They also provide evidence against the claim made by Girshick et al. [18] by showing that halting the learning in the transferred layers can greatly reduce the performance. Lastly, they attempt to see if the effect is still present when more dissimilar datasets are used relative to the source. They do this by separating the dataset into man-made and natural subsets. They find this delivers less improvement compared to the original split. However, this split does not quite capture visual dissimilarity correctly. Since CNNs operate on visual information, more distant tasks should be more visually dissimilar. Consider the case of lemon, tennis ball, and microwave (all classes in the dataset). Visually, a tennis ball and a lemon are similar because of similar shape and color. However, microwave is considered closer to tennis ball than lemon under the categorical man-made definition. The dataset we present better encapsulates visual dissimilarity.

Another question not currently addressed in the literature is the effect of fine-tuning on ensembles of CNNs. The top performing entries of each year's ILSVRC have all used ensembles of CNNs [1, 2, 12, 13]. Any given CNN may overfit to some particular class or classes of data. Because of the stochastic nature of SGD, however, different CNNs trained on the same dataset are unlikely to all overfit exactly the same way. By averaging the output of each individual CNN, the propensity to overfit is reduced, leading to an overall model that is more general [22]. The next section explains the experimental framework used to address the various gaps in the literature in the area of parameter fine-tuning.

3 Experimental framework and dataset

Herein we provide an exposition of an experimental framework and dataset to investigate transfer learning and parameter fine-tuning. To give context to the reader, Fig. 1 shows a simple example of parameter fine-tuning, depicting a network trained on a source task (left) to be applied to a network trained on a target task (right). The source task network, with green nodes, proceeds through training with learned weights represented by blue lines and mapped to two outputs. To transfer the network, a layer is cut off (in this case only the output layer, but not necessarily so). However, due to the mismatched nature of the outputs both in classification and number, the weights in the final remaining layer must be reinitialized, which is represented in the figure by red lines. After transfer, the learned weights optimize SGD to the target task with three outputs.

To investigate gaps in the topic of parameter fine-tuning, this paper addresses three open questions:

1. Should learning be reduced in transferred layers?
2. What type of dataset serves as the best source task?
3. Do ensembles of fine-tuned CNNs outperform ensembles of randomly initialized CNNs?

In order to answer these questions, we assembled a dataset that aggregates seven different fine-grained datasets, described in Table 1 with a breakdown of the data's characteristics. Each dataset has one major classification and a number of minor classification labels. For example, the plane dataset has a major class label of simply plane. It has 78 minor class labels such as Boeing 737 and Spitfire. We refer to the major classes as generalist (G) and the minor as classes as high-fan (HF) for the way they 'fan out' the dataset.

This allows us to create several configurations using the same aggregated dataset. When a network is trained on the entirety of the dataset, it is referred to as a superset network. Depending on how the data are labeled, the superset network may either be a generalist or a high-fan network. A generalist classifies all images at the major class level (e.g., plane vs. bird). A high-fan network classifies everything at the minor class level (e.g., 737 vs. Spitfire vs. Cardinal vs. Crow). Generalist and high-fan networks also have a variant where a single subset is removed. For example, there are the generalist-without-planes (GWP) and high-fan-without-planes (HFWP). A network trained only on a single subset of data is a specialist network.

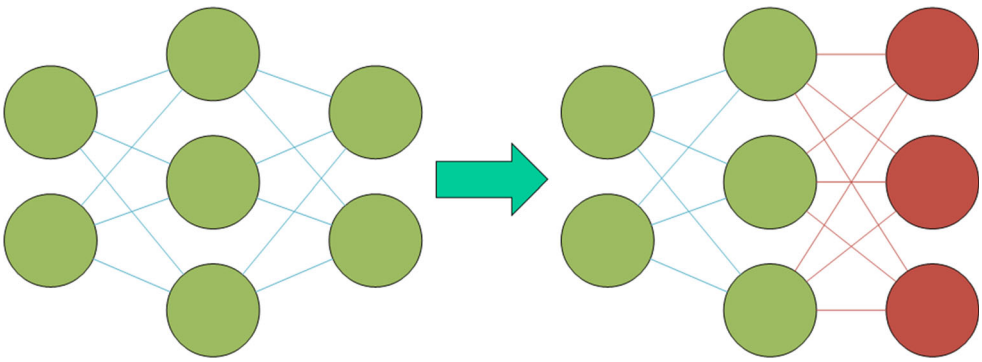

Fig. 1 Fine-tuning visualized, where the network on the left is trained on a task similar to the transfer task. At right the pre-trained network, the output layer is cut off and the final weight layer is transferred to a network with varied outputs (colour figure online)

Table 1 A high-level overview of the multiple class image datasets

Major class	Images	Train	Test	Categories	Median	Source
Vegetable	17,562	13243	4409	24	800	ImageNet [10]
Cat	2392	1795	597	12	200	Oxford Pets [23]
Flower	8189	6180	2009	102	66	Oxford Flowers [24]
Bird	11,788	8872	2916	200	60	CalTech-UCSD Birds [25]
Sign	5886	4424	1462	24	189	KUL Belgian Signs [26]
Dog	71,947	53,982	17,965	111	800	Stanford Dogs[a] [27]
Plane	17,800	13,350	4450	78	100	FGVC [28], Mash[b] [8]
Total	135,654	101,846	33,808	551	91	

[a]Supplemented with data from the given ImageNet synsets

[b]This dataset is not publicly available

Specialist networks always classify at the minor class level. Figure 2 shows an example of the different dataset configurations. It shows the number of images and a notional set of classes in each major class (for readability). This will be discussed further in Sect. 5.

The experimental framework is based on using different initializations for each specialist network. The first is random initialization, which is referred to as a scratch initialization, and is used as a control. The other four come from transferring the weights learned in the superset networks. This framework has a number of inherent advantages. By using 7 different subsets of data, we effectively have 7 different trials to verify results. We can also measure the effect on different types of datasets. For example, the bird subset has a large number of classes but little training data per class, while the dog dataset has fewer

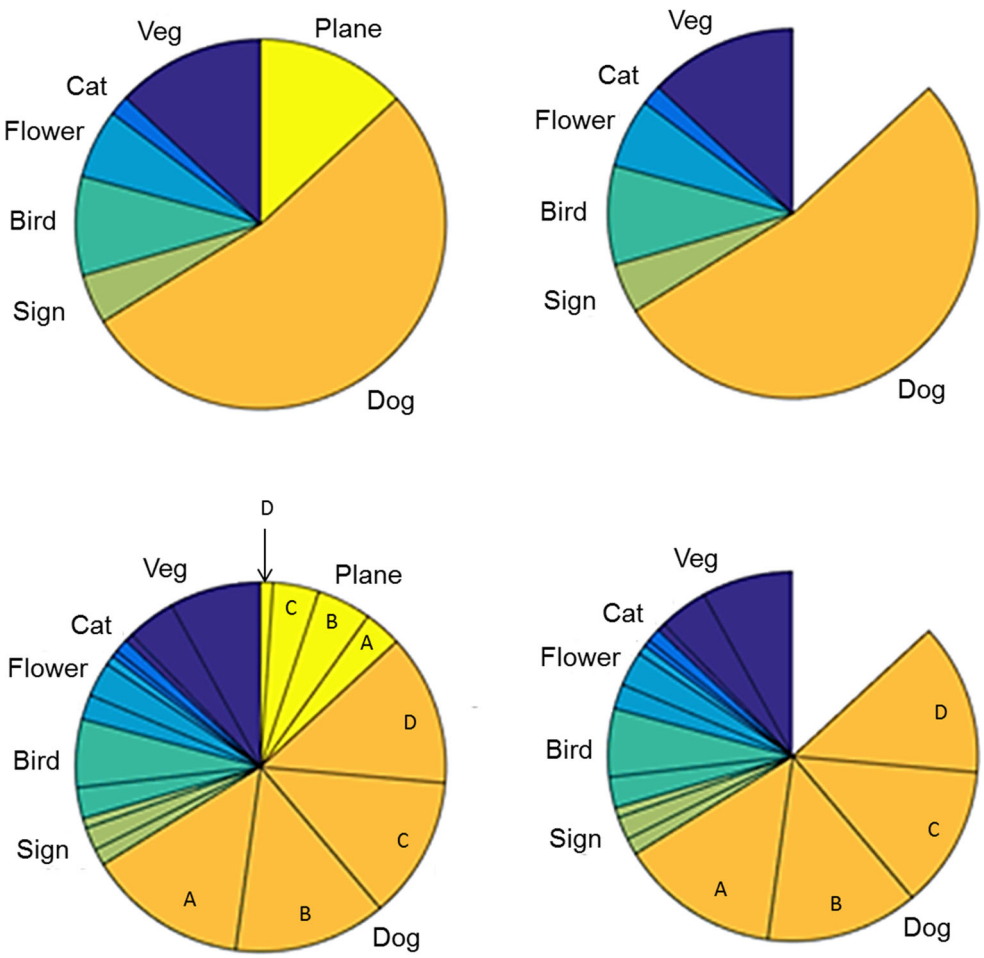

Fig. 2 Different configurations of the multiclass image dataset. At top left a generalist (G) dataset is shown, then a generalist dataset without planes (GWP, top right). At bottom left, a high-fan (HF) dataset, depicting notional subclasses, finally, high-fan-without-planes (HFWP) is shown at bottom right

classes and a much larger amount of training data per class. Data are split 75% for training and 25% for testing. Test data are never used for training on any network. In order to normalize the effect of transfer learning and make the effect comparable across multiple datasets, we use the following formula to measure the error reduction rate:

$$\text{Reduction rate} = \frac{1 - (\text{Scratch errors} - \text{transfer errors})}{\text{Scratch errors}}. \quad (1)$$

There are many parameters that can be adjusted when it comes to training CNNs. Typically, one would tune each parameter to find the best set for their particular problem. However, we are interested only in the effect of fine-tuning. In order to isolate this effect, we use the same set of hyperparameters for all networks. This likely leads to suboptimal performance for particular datasets, but optimal performance is not our goal. Unless otherwise noted, the parameters described below are used across all experiments.

The basic network architecture to be used for this experiment is the same one as in AlexNet [1]. AlexNet has 5 convolutional layers and three fully connected layers attached to a SoftMax layer at the output that generates a probability distribution of all possible classes. While other architectures have been used to achieve better results on the ILVSRC challenge, AlexNet is used for several reasons. First, it is a well-known architecture and has become a standard architecture for experiments on CNNs themselves [3]. Second, even with high-powered GPUs, training a CNN takes days, and more complicated models take longer (the authors of VGG reported that training took 2-3 weeks on similar hardware [13]). Given the number of CNNs that need to be trained for this experiment, time is a nontrivial factor.

There still remains a number of network hyperparameters to be defined. Many of these hyperparameters were chosen by experimenting until the network began to train effectively. Large minibatches can lead to slow convergence, while smaller minibatches may lead to suboptimal convergence. Minibatch size was set to 256. The learn rate was initially set to .001. It was reduced by half every 25,000 training iterations. It is unlikely that this is the best learning strategy for every dataset. However, the goal of this research is not to find the best possible results for each dataset; rather we are trying to study the effect of fine-tuning, so we standardized on a schedule of learning rate multipliers and number of training iterations to measure affects across the datasets. Tweaking the learning strategy for every dataset would multiply the amount of training required, which would have made an experiment on this scale unfeasible due to time constraints. It takes a 3-4 days to capture results on a particular setting. Most networks converged to an optimal solution in less than 100,000 training iterations and then began to overfit; however, for the sake of consistency and thoroughness, all networks were trained for 200,000 iterations. Training consisted of 100 iterations followed by a complete test against the entire validation set. When a network's accuracy is reported, it is the model that had the best accuracy on the validation set. However, the best model was never used for transferring. The model saved at 200,000 iterations was always the source model. This is in line with the research done by Azizpour et al. [21], which found that early stopping was less beneficial than overfitting, although the benefit of overfitting diminishes beyond 200,000 iterations.

All experiments were done using Berkeley Caffe [29]. Caffe was chosen because it supports GPU-accelerated training which makes research on this scale feasible. It also makes it simple to transfer learned weights. Furthermore, it is very popular and well supported by the community. It is expected that the same results would be achieved regardless of deep learning framework used. The main difference is the interface and underlying implementation, not the algorithm.

4 Optimizing learn rate

The first experiment was designed to evaluate the claim by Girshick et al. [18] that there is some value in reducing the learning rate during fine-tuning. To test this claim, we applied a learn rate multiplier to the transferred layers. By varying our learn rate multiplier from 0 to 1, we can measure the effect of preserving the initialization. Using a multiplier of 0 freezes the learning in the transferred layers and absolutely preserves the initialization. We increase the multiplier to 1 in .2 increments. By the time we have a multiplier of 1, we have effectively eschewed the learn rate multiplier and only use the global learn rate. A network with no transferred layers is trained from scratch to serve as a baseline comparison. For this experiment, the transferred weights came from pretraining on the generalist network.

Using the planes subset as the target dataset, Fig. 3 shows the result that using a learning rate multiplier is suboptimal to simply using the global learning rate. Completely freezing the initialization led to an accuracy of 18.584%. It was omitted from Fig. 3 in order to prevent compressing the scale of the rest of the results shown in the figure. Other than that, all other fine-tuned networks outperformed the scratch initialization. The global learning rate performed the best overall. This contradicts the claim of Girshick for the situation where the classification task is aimed at a straightforward labeling of the designated dataset.

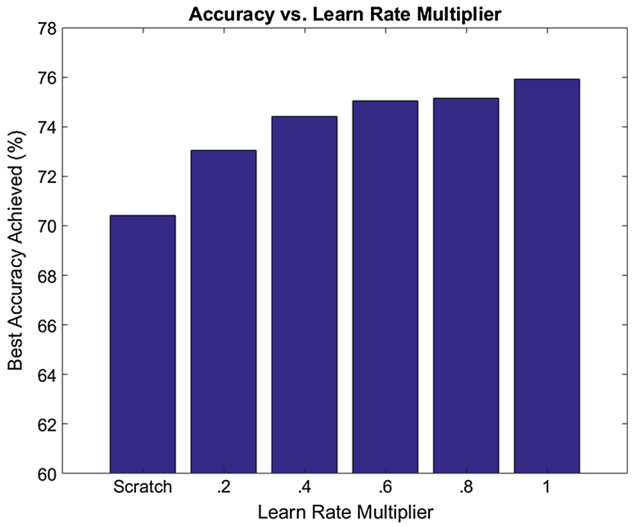

Fig. 3 The results from varying the learn rate multiplier in the transferred layers using the planes subset of the dataset. The scratch network is the control, and the frozen trial was omitted for sake of clarity

Recalling the mechanisms of training a CNN that involve minimizing an optimization problem, the function to be minimized is the cross-entropy loss function of the network. Since cross-entropy loss is a nonconvex function, it may have many local minima that are different from the global minima. Relying on a greedy algorithm like stochastic gradient descent causes convergence to the local minima rather than the global minima. By choosing a better initialization, better minima may be found. Reducing the learn rate of the network after transferring weights does nothing to affect the initialization; if anything, it prevents the valley of the local minima from being efficiently traversed. For this reason we recommend the global learning rate for straightforward labeling tasks. Attempting to interpret Girshick's claim another way, it may be possible to achieve higher classification accuracy if a secondary classification task is to be pursued. We hope to approach that answer in part in the next section.

5 Analyzing effect of the source task

The next experimental question analyzes what type of dataset may serve as the best source for the labeling task, in other words, what data should be used in pretraining. An open question for fine-tuning is which dataset should be chosen for pretraining. As mentioned earlier, our dataset better allows us to measure the effect of source task. Using our dataset, we can control for factors such as the type of source task, the visual similarity of the source task, and the amount of training data in the target task. For each major class in the dataset, we train a specialist network with 5 different initializations. The first is the scratch or random initialization. This serves as the control and acts as the baseline against which all other initializations are compared. The remaining initializations come from transferring weights trained on different configurations of the entire dataset. They are the generalist, high-fan, generalist-without, and high-fan-without datasets where the without network has had the major class of the target task removed. These serve as a more visually distant source task compared to the full dataset. With these different initializations, we can measure the effect source task has on fine-grained image classification as well as the effect that visual similarity has. Furthermore, by repeating this experiment on each major class subset, we can see the effect of increased amounts of training data, starting with Fig. 4, described next.

Figure 4 shows the results from the first five specialist networks. All networks showed an improvement over random initialization regardless of which source was used for fine-tuning. The benefit varied from task to task. For example, both the bird and flower datasets have similar amounts of data in the training set. However, the flower network had a much larger increase in both absolute and relative terms of accuracy compared to the bird dataset. In general, the more distant high-fan-without and generalist-without networks underperformed their more visually similar counterparts. The one exception to this trend is the high-fan-without initialization for the vegetable network was the highest performing initialization for that specialist network.

Figure 5 shows the results from the dog and sign networks, which seem to be the outliers. The dog specialist network is noteworthy because the high-fan-without initialization actually underperformed the random scratch initialization. In the 28 fine-tuned networks trained for this experiment, this was the only instance of this happening. The other initializations also provided very little benefit over random initialization, although they did provide some benefit. The results for the sign specialist network are also shown. These are noteworthy due to the high performance of the scratch network. It was not realized at the time of data collection, but as of 2013 the signs dataset is considered solved [30]. Because it is less difficult and has little room for improvement, this type of dataset is a poor choice for incorporation and is not recommended for transfer learning.

Figure 6 shows the normalized results for all 7 subsets of data. Overall, the high-fan initialization was most often the best choice, turning out to be the top performing initialization 3 out of 7 times. Table 2 shows the raw data for each network as well as the mean of the data. The mean also indicates that the high-fan initialization generally leads to the best performance. However, it was noted that the

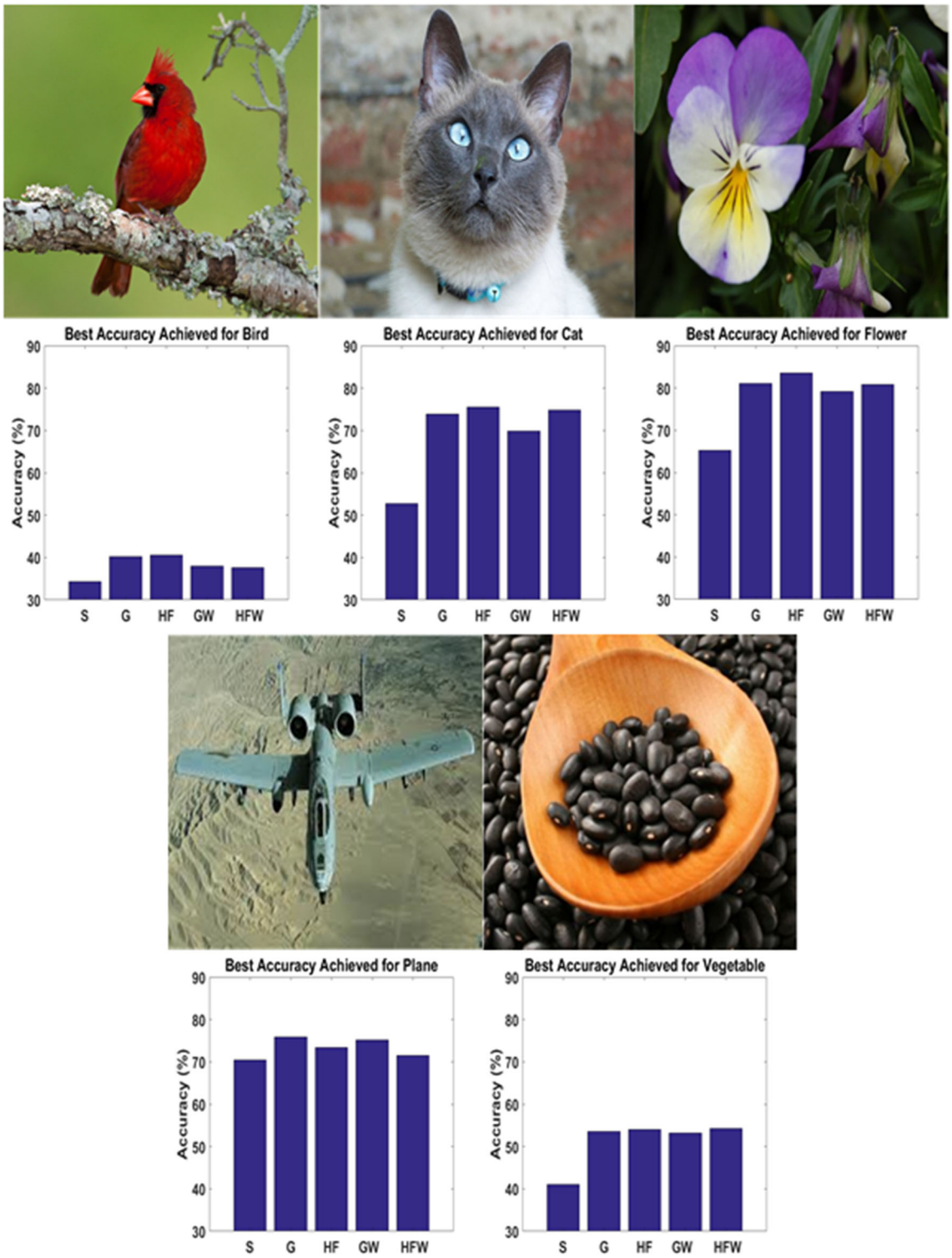

Fig. 4 The results from the first five specialist networks. From left to right: bird, cat, flower, plane, and vegetable. A sample image from each dataset is shown above the graph for that major class

mean indicates that the generalist was the worst performing initialization. This is due to the anomalous behavior of the sign specialist network, where all other initializations outperform the generalist initialization. In no other dataset does the generalist-without initialization outperform the generalist initialization. When the mean is recomputed without the results from the sign dataset, the high-fan networks remain the best initialization and the generalist no

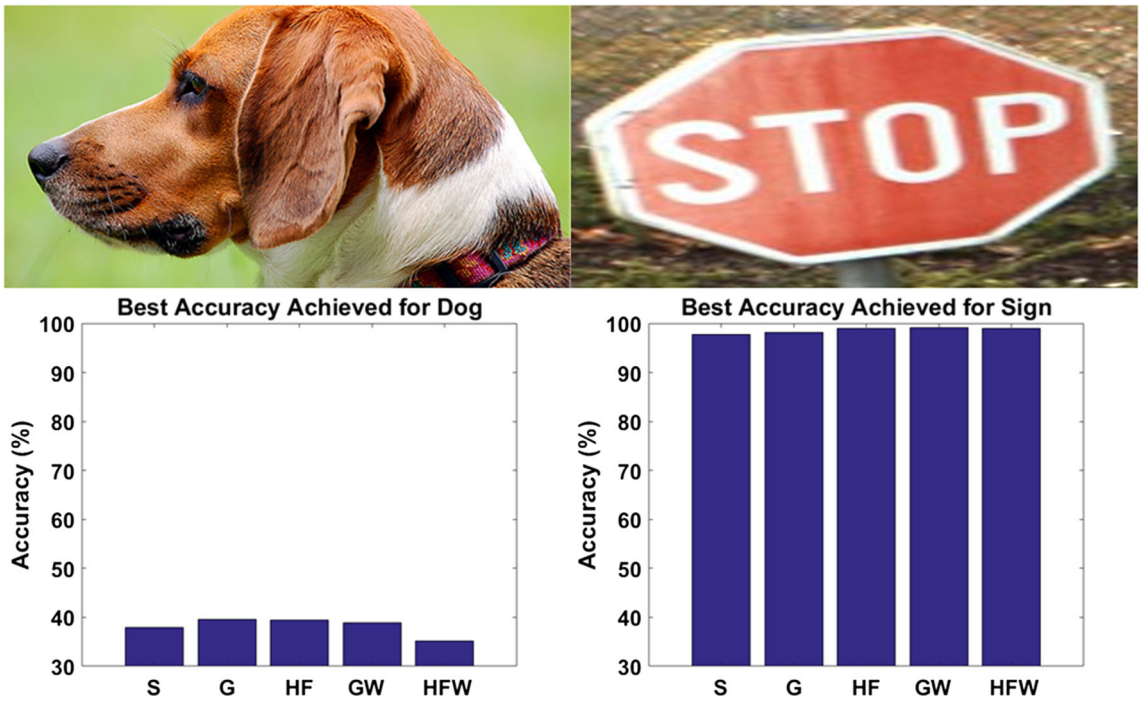

Fig. 5 The results for two outlier specialist networks, dog and sign. A sample image from each dataset is shown above the graph for that major class

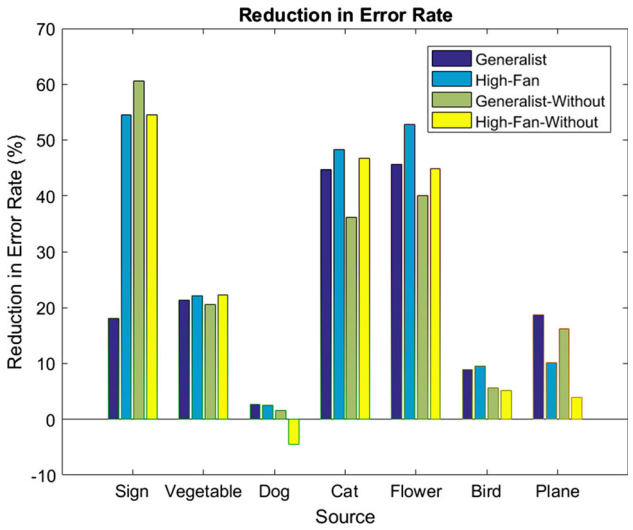

Fig. 6 The error reduction rate of every different datasets. This method normalizes the results and allows us to compare the effect between datasets

longer is seen as underperforming the generalist-without networks.

The results also show that the more visually dissimilar without networks tend to underperform their more similar cousins. This demonstrates that more visually similar tasks should be chosen as the source task. However, there doesn't seem to be a clear connection between the amount of training data available in the target task and the effect on parameter fine-tuning. Both the flower and bird datasets had very little training data available. The best flower initialization leads to an error reduction rate of 52.73%, while the best bird initialization only had an error reduction rate of 9.60%. Meanwhile, the dog dataset, which had the most training data, benefited the least from parameter fine-tuning. The vegetable dataset, which had a similar amount of training data per class but far fewer classes, had a more typical benefit.

6 Ensembles of transfer networks

Ensembles of classifiers have long been known to outperform individuals classifiers because the propensity of any single model to overfit is reduced [5]. Ensembles of CNNs work by simply passing the same image through each individual CNN and taking the mean of each probability distribution to obtain the ensemble's probability distribution of the output. Since fine-tuned networks tend to outperform networks trained from scratch, it seems intuitive that an ensemble of fine-tuned networks would outperform an ensemble of scratch networks. In order to test this, we trained an additional three scratch networks for each major class (except for the sign dataset). This allows direct comparison of 4 scratch networks to the four fine-tuned networks trained in the above experiment. Figure 7 shows the results for the plane specialist networks. As expected,

Table 2 The reduction in error rate for each dataset

Dataset	Generalist (%)	High-fan (%)	Generalist w/out (%)	High-fan w/out (%)
Sign	18.1657	54.5414	60.6114	54.5414
Vegetable	21.2968	22.0706	20.5247	22.3421
Dog	2.7095	2.4938	1.5842	− 4.4402
Cat	44.6799	48.2259	36.1694	46.8075
Flower	45.7029	52.7343	39.9724	44.8423
Bird	8.9763	9.6034	5.5840	5.1137
Plane	18.6788	10.1740	16.2494	3.9466
Mean	22.871	28.5491	25.8136	24.7362
Mean w/out Sign	23.674	24.217	20.014	19.769

the ensembles produced fewer errors than the best model individually. However, it was surprising to see that the best fine-tuned network outperformed the entire ensemble of scratch networks. In fact, the ensemble of scratch networks only outperformed one of the fine-tuned networks. This is particularly interesting because training and deploying an ensemble of 4 CNNs requires 4 times as many computational resources, whereas fine-tuning a single network at most requires only twice the training time and no additional resources for deployment.

Figure 8 shows the results for all major classes. The trend observed in the plane dataset is repeated in all other major classes with one exception. The dog subset, which benefited very little from parameter fine-tuning, has the ensemble of scratch networks outperform the best fine-tuned network. Still, the ensemble of fine-tuned networks outperformed the ensemble of scratch networks, though. Recall that the high-fan-without initialization underperformed the scratch network. It was thought that removing this network from the ensemble might improve the performance of the ensemble. However, doing this caused the overall accuracy to drop from 42.4 to 42.21%. Ensembles

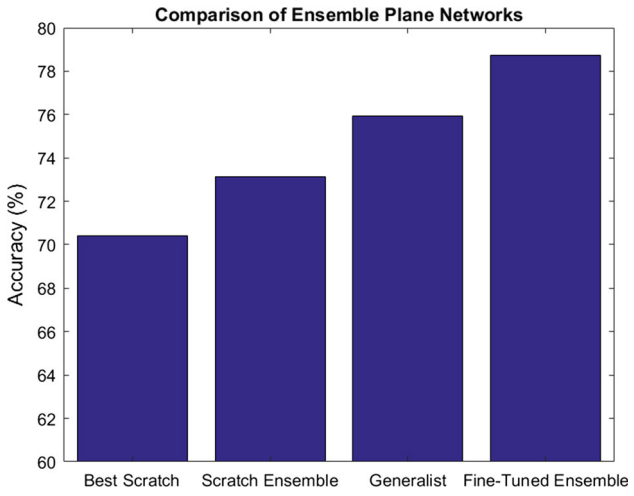

Fig. 7 Comparison of ensembles for the plane subset. The best fine-tuned network alone outperforms the ensemble of scratch networks

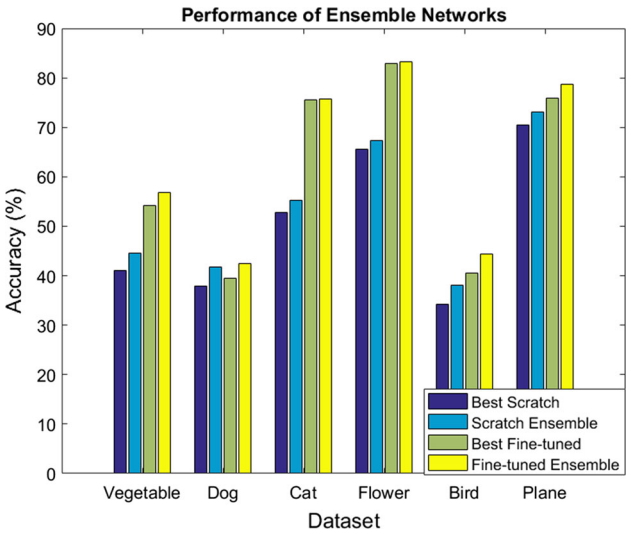

Fig. 8 The accuracy of the best scratch and fine-tuned networks compared to their respective ensembles

of CNNs are more accurate than the individual networks that comprise them. By improving the accuracy of the individual networks through parameter fine-tuning, it seems obvious that the performance of the ensemble would improve as well. Our results confirm this assumption.

7 Conclusion and future work

This paper makes several important contributions. First, we find that contrary to the assertion of Girshick [18], there is no benefit to reducing the learning in a fine-tuned network. The initialization provides a benefit but the initialization is not better that full training on the target. Next, in the study of different source tasks, we observed that pretraining on a dataset with a wide variety of finely grained images provides a better source task than a smaller variety of more general classes. Furthermore, pretraining on a dataset with no overlap with the target dataset provides less of a benefit than a more visually similar source dataset. The amount of training data in the

target task seems to have little correlation with the benefit provided by parameter fine-tuning. Lastly, we study the effect of fine-tuning ensembles of networks. As expected, an ensemble of fine-tuned networks outperforms an ensemble of randomly initialized networks. However, it also seems to be the case that a single fine-tuned network can outperform an entire ensemble of randomly initialized networks. The results demonstrate that parameter fine-tuning almost always leads to an improvement in image classification accuracy. Given these results, we see no reason not to utilize parameter fine-tuning. The only possible downside is the increased training time required for pretraining. However, if using an existing network architecture with available trained models, there may be no increase in training time. Many applications, such as automatic navigation of unmanned vehicles, require a high degree of accuracy in order to rely on a computer vision solution. This technique may help state-of-the-art solutions reach their required thresholds.

There are several directions in which this work could extend. An obvious example would be to apply parameter fine-tuning to a production system to demonstrate that the effect is valid in real-world scenarios. One would be to find the optimal ensemble composition. For example, would the plane specialist ensemble perform better with four separately trained generalist initializations or does the variety of four different initializations produce the benefit? Another area would be to combine our methodology with that of Azizpour et al. [20] to measure the effect of source task on other computer vision problems. Yet another interest area would be to apply our methodology a new architecture such as GoogLeNet to see if our results scale to a much larger CNN [2]. These and certainly other works may extend from this type of research. The value of doing so will continue to refine the theory behind convolutional neural networks since the field is so new. Insights gained from these experimental observations may allow for stronger mathematical constructs and insights to be established. In the meantime, they help inform researchers on best practices for training and pretraining networks for image classification.

Acknowledgements This work was sponsored by the Vehicles Directorate of the Air Force Research Laboratory. The views expressed in this article are those of the author and do not necessarily reflect the official policy or position of the Air Force, Defense Department or the U.S. Government.

References

1. Krizhevsky A, Sulskever I, Hinton GE (2012) ImageNet classification with deep convolutional neural networks. In: Advances in neural information processing systems, pp 1–9
2. Szegedy C, Liu W, Jia Y, Sermanet P (2014) Going deeper with convolutions. arXiv Prepr. arXiv 1409.4842
3. Yosinski J, Clune J, Bengio Y, Lipson H (2014) How transferable are features in deep neural networks? In: Advances in neural information processing systems (Proceedings NIPS), vol 27, pp 1–9
4. Fukushima K, Miyake S (1982) Neocognitron: a new algorithm for pattern recognition tolerant of deformations and shifts in position. Pattern Recognit 15(6):455–469
5. LeCun Y, Boser B, Denker JS, Henderson D, Howard RE, Hubbard W, Jackel LD (1990) Handwritten digit recognition with a back-propagation network. In: Advances in neural information processing systems, vol 2
6. LeCun Y, Bottou L, Bengio Y (1997) Reading checks with multilayer graph transformer networks. In: International conference on acoustics, speech, and signal processing, pp 151–154
7. LeCun Y, Bengio Y, Hinton GE (2015) Deep learning. Nature 521(7553):436–444
8. Mash R, Becherer N, Woolley B, Pecarina J (2016) Toward aircraft recognition with convolutional neural networks. In: National aerospace and electronics conference
9. Strigl D, Kofler K, Podlipnig S (2010) Performance and scalability of GPU-based convolutional neural networks. In: 2010 18th Euromicro conference on parallel, distributed network-based process, pp 317–324
10. Deng J, Dong W, Socher R, Li L-J, Li K, Fei-Fei L (2009) ImageNet: a large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition, pp 2–9
11. Hinton GE, Srivastava N, Krizhevsky A, Sutskever I, Salakhutdinov RR (2012) Improving neural networks by preventing co-adaptation of feature detectors. arXiv e-prints, pp 1–18
12. Sermanet P, Eigen D, Zhang X, Mathieu M, Fergus R, LeCun Y (2013) OverFeat: integrated recognition, localization and detection using convolutional networks. arXiv Prepr. arXiv, p 1312.6229
13. Simonyan K, Zisserman A (2014) Very deep convolutional networks for large-scale image recognition. In: ImageNet Challenge, pp 1–10
14. Howard AG (2013) Some improvements on deep convolutional neural network based image classification. arXiv Prepr. arXiv1312.5402, pp 1–6
15. Krähenbühl P, Doersch C, Donahue J, Darrell T (2015) Data-dependent initializations of convolutional neural networks. In: International conference on computer vision, pp 1–12
16. Lowe DG (1999) SIFT. Comput Vis 2:1150–1157
17. Bay H, Tuytelaars T, Van Gool L, Leonardis A, Bischof H, Pinz A (2006) SURF: speeded up robust features. Comput Vis 3951:404–417
18. Girshick R, Donahue J, Darrell T, Malik J (2014) Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 580–587
19. Uijlings JRR, Van De Sande KEA, Gevers T, Smeulders AWM (2013) Selective search for object recognition. Int J Comput Vis 104(2):154–171
20. Sharif Razavian A, Azizpour H, Sullivan J, Carlsson S, (2014) CNN features off-the-shelf: an astounding baseline for recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition workshops, pp 806–813

21. Azizpour H, Sharif Razavian A, Sullivan J, Maki A, Carlsson S, (2015) From generic to specific deep representations for visual recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pp 36–45
22. Giacinto G, Roli F (2001) Design of effective neural network ensembles for image classification purposes. Image Vis Comput 19(9–10):699–707
23. Parkhi OM, Vedaldi A, Zisserman A, Jawahar CV (2012) Cats and dogs. In: 2012 IEEE conference on computer vision and pattern recognition, pp 3498–3505
24. Nilsback M-E, Zisserman A (2006) A visual vocabulary for flower classification. In: Computer vision and pattern recognition, pp 1447–1454
25. Welinder P, Branson S, Mita T, Wah C (2010) Caltech-UCSD birds 200. CalTech 200:1–15
26. Timofte R, Zimmermann K, Van Gool L (2014) Multi-view traffic sign detection, recognition, and 3D localisation. Mach Vis Appl 25(3):633–647
27. Khosla A, Jayadevaprakash N, Yao B, Fei-Fei L (2011) Novel dataset for fine-grained image categorization: Stanford dogs. In: First work. Fine-grained visual categorization. IEEE Conference on computer vision and pattern recognition
28. Maji S, Rahtu E, Kannala J, Blaschko M, Vedaldi A (2013) Fine-grained visual classification of aircraft. TechReport
29. Jia Y, Shelhamer E, Donahue J, Karayev S, Long J, Girshick R, Guadarrama S, Darrell T (2014) Caffe: convolutional architecture for fast feature embedding. In: Proceedings of ACM international conference on multimedia—MM'14, pp 675–678
30. Mathias M, Timofte R, Benenson R, Van Gool L (2013) Traffic sign recognition—How far are we from the solution? In: 2013 international joint conference on neural networks (IJCNN), pp 1–8